# Placement Exercise

(a) Run vpr using ***-seed*** option with seed values 1, 7 and 15. Tabulate the results for each benchmark circuit. Show initial BB cost, final BB cost and percentage reduction in BB cost. Also show the approximate run time. Use default values for all other options, e.g.

***vpr apex2.net 4lut_sanitized.arch apex2.p.seed5 -place_only -seed 5 > run_seed5 &***

a) -seed <int>: Sets the initial random seed used by the placer. Default: 1.

Using the -seed option with seed values 1, 5, 7 and 15, following results were obtained with vpr:

| Benchmark circuit (No. Of netlist primities) | Seed value | bb_cost | | Percentage Reduction in bb_cost | Runtime (mm:ss) | Placement cost | | timing_cost | | delay_cost | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Initial | Final | | | Initial | Final | Initial | Final | Initial | Final |
| Apex2 (1116) | 1 | 894.188 | 280.12 | 68.67% | 0:42.59 | 0.996 | 0.879 | 1.27e-4 | 6.23e-6 | 1.41e-4 | 7.27e-5 |
| | 5 | 903.734 | 280.641 | 68.94% | 0:42.98 | 0.994 | 0.836 | 1.27e-4 | 3.61e-6 | 1.40e-4 | 7.23e-5 |
| | 7 | 894.452 | 279.275 | 68.78% | 0:43.17 | 0.980 | 0.851 | 1.26e-4 | 6.04e-6 | 1.39e-4 | 7.16e-5 |
| | 15 | 904.57 | 279.799 | 69.07% | 0:43.25 | 1.00682 | 0.893 | 1.27e-4 | 5.19e-6 | 1.40e-4 | 7.28e-5 |
| Frisc (3291) | 1 | 2271.44 | 587.227 | 74.15% | 1:51.29 | 0.999 | 0.939 | 1.21e-4 | 3.87e-6 | 3.29e-4 | 1.39e-4 |
| | 5 | 2275.88 | 589.02 | 74.12% | 1:54.66 | 1.00763 | 0.941 | 1.21e-4 | 6.37e-6 | 3.29e-4 | 1.36e-4 |
| | 7 | 2252.98 | 587.466 | 73.92% | 1:53.71 | 0.991 | 0.941 | 1.18e-4 | 6.11e-6 | 3.22e-4 | 1.44e-4 |
| | 15 | 2249.87 | 593.212 | 73.63% | 1:53.10 | 0.991 | 0.947 | 1.18e-4 | 7.12e-6 | 3.22e-4 | 1.36e-4 |
| s38417 (4888) | 1 | 5835.43 | 687.198 | 88.22% | 5:18.81 | 0.995 | 0.572 | 4.60e-4 | 6.77e-6 | 6.94e-4 | 1.85e-4 |
| | 5 | 5828.23 | 689.025 | 88.18% | 4:49.80 | 1.00278 | 0.577 | 4.65e-4 | 7.98e-6 | 7.05e-4 | 1.83e-4 |
| | 7 | 5827.58 | 694.011 | 88.09% | 4:55.78 | 0.986 | 0.604 | 4.51e-4 | 3.79e-6 | 6.83e-4 | 1.79e-4 |
| | 15 | 5790.68 | 686.778 | 88.14% | 5:07.13 | 0.987 | 0.567 | 4.53e-4 | 3.68e-6 | 6.84e-4 | 1.87e-4 |

Table 1

Inferences:

* Since, seed value determines the initial random placement, the initial bb_cost varies with the seed value for each benchmark circuit.

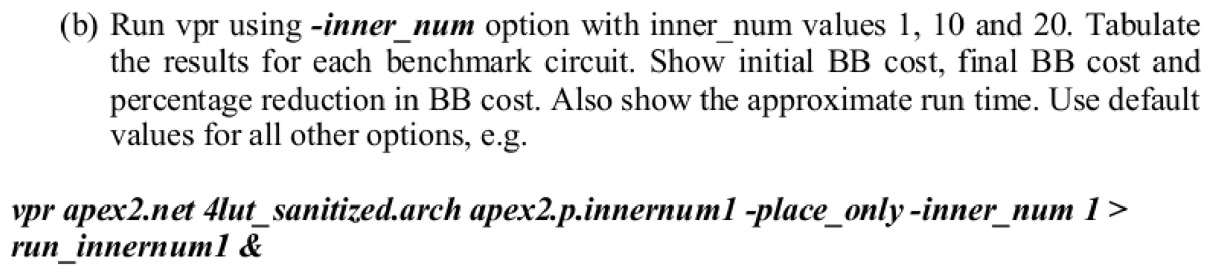
* “Percentage reduction in bb_cost” (highlighted in green) and “final bb_cost” (highlighted in turqoise) seem to be independant of the seed value for the 3 benchmark circuits.

* Final bb_cost achived is very close (Low standard deviation) for the four seed values for each benchmark circuit:

| Benchmark Circuit | Average final bb_cost | Standard deviation |
|---|---|---|
| apex2 | 279.96 | 0.57 |
| frisc | 589.23 | 2.77 |
| s38417 | 689.25 | 3.32 |

* Percentage reduction achieved in bb_cost increases with design size i.e number of netlist primitives

* Runtime for the vpr seems to be independant of the seed value but depends on the design size (as expected).



(b) Run vpr using ***-inner_num*** option with inner_num values 1, 10 and 20. Tabulate the results for each benchmark circuit. Show initial BB cost, final BB cost and percentage reduction in BB cost. Also show the approximate run time. Use default values for all other options, e.g.

```
vpr apex2.net 4lut_sanitized.arch apex2.p.innernum1 -place_only -inner_num 1 >
run_innernum1 &
```

b) -inner_num <float>: The number of moves attempted at each temperature is inner_num times the total number of blocks 4/3 in the circuit. The number of blocks in a circuit is the number of pads plus the number of clbs. Changing inner_num is the best way to change the speed/quality tradeoff of the placer, as it leaves the highly-efficient automatic annealing schedule on and simply changes the number of moves per temperature. Default: 10. Note: specifying -inner_num 1 will speed up the placer by a factor of 10 while typically reducing placement quality only by 10% or less (depends on the architecture). Hence users more concerned with CPU time than quality may find this a more appropriate value of inner_num.

Using the -inner_num option with inner_num values 1, 10, and 20, following results were obtained with vpr:

Table 2

| Benchmark circuit (No. Of netlist primities) | Inner num value | bb_cost | | Percentage Reduction in bb_cost | Runtime (mm:ss) | Placement cost | | timing_cost | | delay_cost | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Initial | Final | | | Initial | Final | Initial | Final | Initial | Final |
| Apex2 (1116) | 1 | 894.188 | 290.454 | 67.51% | 00:04.90 | 0.996 | 0.993 | 1.27e-4 | 7.05e-6 | 1.41e-4 | 6.88e-5 |
| | 10 | 894.188 | 280.12 | 68.67% | 00:44.05 | 0.996 | 0.879 | 1.27e-4 | 6.23e-6 | 1.41e-4 | 7.27e-5 |
| | 20 | 894.188 | 275.449 | 69.19% | 01:24.26 | 0.996 | 0.897 | 1.27e-4 | 3.28e-6 | 1.41e-4 | 7.26e-5 |
| Frisc (3291) | 1 | 2271.44 | 681.733 | 69.99% | 00:12.53 | 0.999 | 0.997 | 1.21e-4 | 4.63e-6 | 3.29e-4 | 1.55e-4 |
| | 10 | 2271.44 | 587.227 | 74.15% | 01:54.53 | 0.999 | 0.939 | 1.21e-4 | 3.87e-6 | 3.29e-4 | 1.39e-4 |
| | 20 | 2271.44 | 586.742 | 74.17% | 03:47.84 | 0.999 | 0.959 | 1.21e-4 | 6.14e-6 | 3.29e-4 | 1.38e-4 |
| s38417 (4888) | 1 | 5835.43 | 726.34 | 87.55% | 00:34.60 | 0.995 | 0.960 | 4.60e-4 | 1.29e-5 | 6.94e-4 | 1.70e-4 |
| | 10 | 5835.43 | 687.198 | 88.22% | 04:54.88 | 0.995 | 0.572 | 4.60e-4 | 6.77e-6 | 6.94e-4 | 1.85e-4 |
| | 20 | 5835.43 | 671.954 | 88.48% | 09:11.89 | 0.995 | 0.549 | 4.60e-4 | 5.31e-6 | 6.94e-4 | 1.86e-4 |

Inferences:

* Since, number of moves attempted at each temperature is a multiple of inner_num, hence, increasing inner_num value increases the runtime of the simulated annealing algorithm. As can be seen from table 2 above, runtime is maximum for inner_num value of 20 for each benchmark circuit. Similarly, the minimum runtime was seen with inner_num value of 1.

* Also, the quality of results i.e. the bb_cost achieved, improves with higher number for inner_num. This is because the number of moves attempted at each temperature increases with increase in inner_num value. For each benchmark circuit, the best placement was achieved (lowest bb_cost) for inner_num = 20

* As was the case with different seed values, the maximum runtime is seen for design with maximum number of blocks i.e. s38417

---

(c) Run vpr using all default values with and without the *-fast* option. Tabulate and compare the results for each benchmark circuit, with and without –**fast** option. Show initial BB cost, final BB cost and percentage reduction in BB cost. Also show the approximate run time. For **-fast** option, use default values for all other options, e.g.

***vpr apex2.net 4lut_sanitized.arch apex2.p.fast -fast -place_only > run_fast &***

c)

-fast: Sets various placer and router parameters so that a circuit will be placed and routed more quickly, at the cost of some (~10 - 15%) degradation in quality. Note: -fast should probably not be used with the timing-driven router, as it can degrade the circuit speed by over 30%, without significantly speeding the router.

| Benchmark circuit (No. Of netlist primities) | Fast option | bb_cost | | Percentage Reduction in bb_cost | Runtime (mm:ss) | Placement cost | | timing_cost | | delay_cost | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Initial | Final | | | Initial | Final | Initial | Final | Initial | Final |
| Apex2 (1116) | Yes | 894.188 | 290.454 | 67.52% | 0:04.75 | 0.996 | 0.993 | 1.27e-4 | 7.05e-6 | 1.41e-4 | 6.88e-5 |
| | No | 894.188 | 280.12 | 68.67% | 0:42.18 | 0.996 | 0.879 | 1.27e-4 | 6.23e-6 | 1.41e-4 | 7.27e-5 |
| Frisc (3291) | Yes | 2271.44 | 681.733 | 69.98% | 0.12.22 | 0.999 | 0.997 | 1.21e-4 | 4.63e-6 | 3.29e-4 | 1.55e-4 |
| | No | 2271.44 | 587.227 | 74.15% | 1:51.60 | 0.999 | 0.939 | 1.21e-4 | 3.87e-6 | 3.29e-4 | 1.39e-4 |
| s38417 (4888) | Yes | 5835.43 | 726.34 | 87.55% | 0:30.34 | 0.995 | 0.960 | 4.60e-4 | 1.29e-5 | 6.94e-4 | 1.70e-4 |
| | No | 5835.43 | 687.198 | 88.22% | 4:35.98 | 0.995 | 0.572 | 4.60e-4 | 6.77e-6 | 6.94e-4 | 1.85e-4 |

Table 3

Inferences:

* Runtime was reduced to 10-15% of the original runtime when the -fast option is used.

* Although, a slight degradation is seen in the bb_cost achieved.

* Initial placement remained unaffected due to the -fast option.

=> The runtime for the various runs was obtained through the "time" command. vpr tool was run as an argument to the time command.